PRY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/838,552	04/18/2001	Boris A. Babaian	020181004900	8760

20350 7590 07/29/2004

TOWNSEND AND TOWNSEND AND CREW, LLP
TWO EMBARCADERO CENTER
EIGHTH FLOOR
SAN FRANCISCO, CA 94111-3834

EXAMINER	
STEELMAN, MARY J	
ART UNIT	PAPER NUMBER
2122	5

DATE MAILED: 07/29/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/838,552	BABAIAN ET AL.
	Examiner	Art Unit
	Mary J. Steelman	2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *4/18/01, 8/8/01*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-34* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-34* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *18 April 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *8/8/01*

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-34 are pending.

Drawings

2. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they do not include the following reference character(s) mentioned in the description:

FIGURE 1, Call/Return Cache 118, should be 108. See Specification, page 8, line 3.

FIGURE 3, see Specification page 11, line 3, 'very long instruction word format 300'. 300 is not in the drawing.

FIGURE 5, See specification, page 19, lines 33-34. 914 in Figure 5 should be 514.

FIGURE 5, See Specification, page 19, line 6: 'exception handler 932' should be 532.

See Specification, page 20, line 29, exception 940 should be 530.

Corrected drawing sheets are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

3. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) not mentioned in the description:

FIGURE 2: 206 See Specification, page 8, line 34, software layer 208. (Edit Specification to read 206.)

Corrected drawing sheets, or amendment to the specification to add the reference character(s) in the description, are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Information Disclosure Statement

4. IDS received 8 August 2001 has been considered.

Specification

5. Applicant is requested to update the blanks in page 1 of the Specification.

6. Examiner objects to the Abstract exceeding the 150 word limit.

Content of Specification

(j) Abstract of the Disclosure: See MPEP § 608.01(f). A brief narrative of the disclosure as a whole in a single paragraph of 150 words or less commencing on a separate sheet following the claims. In an international application which has entered the national stage (37 CFR 1.491(b)), the applicant need not submit an abstract commencing on a separate sheet if an abstract was published with the international application under PCT Article 21. The abstract that appears on the cover page of the pamphlet published by the International Bureau (IB) of the World Intellectual Property Organization (WIPO) is the abstract that will be used by the USPTO. See MPEP § 1893.03(e).

Claim Objections

7. Claims 2, 5, 17, 19, 20, and 29 are objected to because of the following informalities:

Claim 2, line 3, recites, "correspond to said executable instruction", should be – corresponds to said executable instruction--. Add an 's'/

Claim 5 recites, "The method of claim 5…", should be—The method of claim 1--. Examiner will treat claim 5 as if it recites, "The method of claim 1…"

Claim 17 I) is not a step.

Claim 19, line 3 recites, "…when an exception while…", should be ''when an exception occurs while…-- Add 'occurs'.

Claim 20 c), recites "…invoked the interpretation…", should be –invoking the interpretation…--.

Claim 29: Add a period, '.', at the end of the claim.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

8. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

9. Claim 26 recites the limitation "said next to occur Recovery Point" in lines 2 & 3. There is insufficient antecedent basis for this limitation in the claim.

10. Claim 28 recites the limitation "the wide instruction" in line 2. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 102

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1-34 are rejected under 35 U.S.C. 102(b) as being anticipated by US Patent 5,832,205 to Kelly et al.

Per claim 1:

In a computer system adapted to executing binary translated code, a method for responding to an exception comprising the steps of:

(Col. 1, lines 10-12, "This invention relates to computer systems and more particularly, to methods…")

-defining a plurality of recovery points in the binary translated code, each of said plurality of recovery points comprising an executable instruction;

(Col. 12, lines 59-60, "These updates are chosen by the code morphing software to occur on integral target instruction boundaries." Instruction segments are translated, executed and checked for error.)

-prior to execution of said executable instruction, saving the address of said executable instruction;

(Col. 19, lines 24-30, "the instruction pointer is updated in the target instruction pointer register…software looks up the instruction pointer to the next target instruction…")

-executing subsequent instructions;

(Col. 19, lines 33-36, "…the next set of target instructions is fetched from memory, decoded, translated, optimized, reordered…and executed.")

-detecting an exception;

(Col. 17, lines 47-48,"If a target exception is generated…that exception is detected…")

-invoking an exception handler;

(Col. 17, 49-51, "In response to the detection of the target exception…cause the values retained in the official registers to be placed back into the working registers and any non-committed memory stores in the gated store buffer to be dumped (exception handling invoked)…")

-reconfiguring the state of the computer system to reflect the state of the computer system at said executable instruction;

(Col. 13, lines 32-35, "This requires that the code morphing software provide for saving the state of the target processor", 38-39, "...a return to the beginning of the process being translated...", 41-43, "require that the exception handler return the operation to the next step in the translation after the exception has been handled.", col. 17, lines 51-54, "In response to the detection of the target exception, the code morphing software may cause the values retained in the official registers to be laced back into the working registers and any non-committed memory stores in the gated store buffer to be dumped." (reconfigure state back to previous instruction))

-using run-time information, translating foreign code beginning at an instruction corresponding to said executable instruction;

(Col. 17, lines 59-62, "Placing the values from the target registers (using run-time information) into the working registers may place the address of the first of the target instructions which were running when the exception occurred in the working instruction pointer register." Return to a previously translated instruction using the code translation software to re-translate foreign code after an exception has been detected.)

-determining if said exception re-occurs.

(Col. 18, lines 9-14, "...if a translation of a target instruction is executed without an exception...However, if an exception re-occurs...")

Per claim 2:

-upon resolving the exception, returning to said binary translated code following the recovery point that corresponds to said executable instruction.

(Col. 17, lines 59-62, "Placing the values from the target registers into the working registers may place the address of the first of the target instruction which were running when the exception occurred in the working instruction pointer register.")

Per claim 3:

-terminating execution of said binary translated code if said exception handler is unable to resolve the exception.

(Col. 18, lines 14-20, "However, if an exception re-occurs…when the target exception is generated, the exception will be correctly handled by the target operating system." (Termination is the correct handling of an exception that is unable to be resolved.)

Per claim 4:

-determining said plurality of recovery points during binary translation of foreign code.

(Col. 12, lines 59-60, "These updates are chosen by the code morphing software to occur on integral target instruction boundaries." Recovery points are on instruction boundaries. Instructions are translated and run according to instruction boundaries. In the case of an exception, the instruction pointer is reset to the first instruction of the instruction segment and re-translated.)

Per claim 5:

-defining said plurality of recovery points is performed at run-time.

(Col. 12, lines 59-65, "These updates are chosen by the code morphing software to occur on integral target instruction boundaries (recovery points are at each instruction boundary)....if the primitive host instruction making up a translation of a series of target instructions are run...without generating exceptions...if an exception occurs while processing the host instructions at a point which is not on the boundary (end of a segment of instructions) ...the original state in the target register ...may be recalled...")

Per claim 6:

-instructions comprise a plurality of operations and said defining step further comprises the step of optimizing the sequence of execution of said operations.

(Col. 14, lines 25-28, "The single very long instruction for the VLIW processor illustrated includes each of the more primitive operations (i.e., load, store, integer add, compare, floating point multiply, and branch)...(plurality of operations)", col. 14, line 54, "...optimized the operations...")

Per claim 7:

-optimizing step further comprises the steps of:

(Col. 11, line 49, "...other optimization techniques...")

-maintaining documentation describing intermediate results of said operations;

(Col. 12, lines 37-40, "The additional registers also allow the maintenance of a set of host or working registers for processing the host instructions and a set of target registers to hold the official state of the target processor...", col. 17, lines 1-6, "...a gated store buffer ...is utilized in the hardware of the improved microprocessor to control the transfer of data...", col. 17, lines 27-36, "This continues until a commit command is executed...memory stores in the store buffer generated during execution are moved (intermediate results are maintained / committed) together past thee gate of the store buffer (committed) and subsequently written to memory...by copying the value in the register...")

-committing the values of said plurality of registers to documentation prior to execution of said executable instruction.

(Col. 12, lines 41-44, "The target (or shadow) registers are connected to their working register equivalents through a dedicated interface that allows an operation called 'commit' to quickly transfer the content of all working registers to official target registers...")

Per claim 8:

said invoking step further comprises the steps of:

-recovering the address of said executable instruction;

(Col. 12, lines 41-44, "The target (or shadow) registers are connected to their working register equivalents through a dedicated interface that allows an operation called 'commit' to quickly transfer the content of all working registers to official target registers...")

-determining the location of said set of registers;

(Col. 12, line 42, "The target (or shadow) register are connected to their working register equivalents…")

-invoking a binary translator to re-translate a sequence of foreign code corresponding to said executable instruction and said subsequent instructions, said binary translator adapted to generate an in-order sequence of binary translated code corresponding to said foreign code.

(Col. 17, lines 65-66, "…are retranslated in serial order…")

Per claim 9:

said invoking step further comprises the steps of:

-committing the state of said computer system to said documentation at a next to occur recovery point.

(Col. 17, lines 31-36, "…executed…without error…(committed) and subsequently written to memory…", col. 18, lines 23-24, "…target instruction pointer points to the next of the target instructions. (next to occur recovery point)")

Per claim 10:

In a computer system adapted for executing optimized binary translated code while maintaining precise exception order, said computer system comprising:

(Col. 17, lines 50-51, "In response to the detection of the target exception, the code morphing software may cause the values retained in the official registers to be placed back into the working registers and any non-committed memory stores in the gated store buffer to be dumped…" If an exception is detected, it is retranslated. The exception order is maintained.)

-a processing unit for executing a sequence of instructions, each of said instructions comprising a plurality of operations that may be executed in parallel; each of said operations having at least one operand; said processing unit adapted to detect the occurrence of an exception;
(Col. 14, lines 14-17, "…morph host is a very long instruction word (VLIW) processor which is designed with a plurality of processing channels (execute in parallel)", col. 14, lines 50-60, "…translator portion which decodes…converts those target instructions to the primitive host instructions…optimizes…into VLIW instructions…", col. 14, lines 36-38, "The results (operands) of all of these parallel operations are transferred into a multiported register file", col. 16, lines 17-18, "…a number of optimizations including register renaming may be practiced…", Col. 17, lines 47-49, "If a target exception is generated…that exception is detected…(detect occurrence of an exception)")

-a register set for holding results of said operations and the state of said
computer system as determined by execution of said operations and for storing the value of said at least one operand prior to executing said each of said instructions;
(Col. 16, lines 21-26, "…enhanced set of target registers which include all of the frequently changed registers of the target processor necessary to provide the state of that processor…", col. 16, lines 43-46, "…the official target registers hold the values which would be held by the registers of the target processor for which the application was designed when the first target

instruction was addressed.", col. 17, lines 31-36, "...executed...without error, then the memory stores in the store buffer generated during execution are moved together past the gate of the store buffer (committed) and subsequently written to memory...copying the value in the register...", col. 17, lines 41-44, "...transfer of register state from working registers to official target registers and the transfer of working memory stores to official memory..." Target registers hold the official state and results of operations / operands from prior instruction execution. Working registers/ host registers hold temporary values created during a current execution. If the execution does not produce an exception, the values in the working registers are committed to the target registers.)

-an exception handler adapted to access said register set to recover the value of said operand and to generate an intermediate state representative of the state of said processing unit prior to the detection of the exception.
(Col. 17, lines 47-54, "If a target exception is generated during the running of any translated instruction or instructions, that exception is detected by the morph host hardware of software...may cause the values retained in the official registers to be placed back into the working registers and any non-committed memory stores in the gated store buffer to be dumped..." A rollback to a former committed state.)

Per claim 11:
-means for transferring the contents of said register set to a storage location prior to execution of a subsequent instruction.

(Col. 17, lines 31-35, "When a translation has been executed by the morph host without error, the n the memory stores...written to memory...copying the value in the register (transfer the contents of the register to a storage location prior to executing the next set of instructions)...")

Per claim 12:

-exception handler further comprises a set of documentation identifying a memory location for said contents.

(Col. 17, lines 4-6, "The gated store buffer includes a number of elements each of which may hold the address and data for a memory store operation.")

Per claim 13:

-exception handler further comprises means for recovering from said exception by translating a portion of foreign code corresponding to said each of said instructions.

(Col. 17, lines 59-62, "Placing the values from the target registers into the working registers may place the address of the first of the target instructions which were running when the exception occurred in the working instruction pointer register." Recover the address of the first instruction, before re-translating.)

Per claim 14:

In a computer system adapted for executing optimized binary translated code while maintaining precise exception order of foreign code, a method for recovering from a detected exception comprising the steps of:

-documenting information representative of the state of said computer system;

(Col. 12, lines 37-41, "The additional registers also allow the maintenance of set of host or working registers for processing the host instructions and a set of target registers to hold the official state (information representative of the state) of the target processor for which the target application was created.")

-executing at least a first instruction, said first instruction having a plurality of operations executed in speculative mode;

(Col. 13, lines 47-57, "...some exceptions are generated...and detect a variety of hast and target conditions...but others are used by the code morphing software to detect failure of various speculations...In these cases...using the state saving and restoring mechanisms...causes the target state to be restored to its most recent official version and generates and saves a new translation...which avoids the failed speculation")

-executing a second instruction;

(Col. 12, lines 60-66, "These updates are chosen by the code morphing software to occur on integral target instruction boundaries." If the system successfully translates and executes a section of instructions, then it returns and translates and executes the next section of instructions.)

-detecting whether the execution of said second instruction generated an exception;

(Col. 13, line 51, "...detect failure of various speculations...")

-upon detection of an exception, transferring information to a register set;

(Col. 13, lines 51-57, "...using the state saving and restoring mechanisms...causes the target state to be restored...and saves a new translation...")

-repeating execution of said at least a first instruction;

(Col. 13, line 67 – col. 14, line 2, "...recovery is accomplished by the generation of new translations with different memory operations and different optimizations.")

-transferring program execution to an exception handler.

(Col. 13, lines 59-60, "...exception detection mechanisms (handle exceptions) that in conjunction with the rollback and retranslate method...")

Per claim 15:

-register set comprises a set of general purpose registers.

(Col. 12, lines 31-35, "...a large plurality of additional processor registers...")

Per claim 16:

-said documenting information is stored in a portion of said register set.

(Col. 13, lines 38-39, "...allow the maintenance of a set of host or working registers (hold documenting information until a commit command)...")

Per claim 17:

In a computer system adapted for executing optimized binary translated code while maintaining precise exception processing of foreign code comprising the steps of:

(Col. 13, line 60, "...retranslate method...allow further optimization (optimized binary translation)..." Precise exception processing is maintained by executing a portion of instructions, determining whether an exception exists, then handling the exception or proceeding with additional translations and executions.)

A) Executing a first sequence of instructions in a speculative mode where each of said instructions comprise a plurality of operations which may be executed in parallel;

(FIGs. 6 a-c, Col. 14, lines 14-17, "...morph host is a very long instruction word (VLIW) processor which is designed with a plurality of processing channels. (execute in parallel)", col. 13, line 51, "...detect failure of various speculations..." Translated instructions may be executed in speculative mode. Various speculation exceptions may be detected.)

B) Saving the address of at least one of the instructions in said sequence of instructions in a storage location;

(Col. 16, lines 9-15, "In order to determine the correct state of the register at the time an error occurs, a set of official target registers is provided...to hold the state of the registers of the target processor...", col. 16, lines 23-26, "...registers of the target processor necessary to provide the state of that processor, these include condition control registers and other registers necessary for control of the simulated system." The address of the instructions is saved in the target's set of official registers to use if a rollback is needed.)

C) Saving the status of said computer system in a second storage location each time said at least one of the instructions is executed;
(Col. 16, lines 46-49, "After the morph host has begun executing the translated instruction…the working register hold values determined by the primitive operations of the translated instructions executed to that point. (Second storage location is set of working registers, which hold temporary values, until the values are committed to the target / official registers.)

D) Executing a next to occur instruction in said first sequence of instructions;
(FIG. 8, Col. 16, lines 28-31, "…a translated instruction sequence (next to occur instruction) may include…")

E) Detecting whether the execution of said next to occur instruction generated an exception;
(Col. 17, lines 47-50, "If a target exception is generated during the running of any translated instruction or instructions (next to occur instruction), that exception is detected…")

F) If an exception is detected, recovering said address from said storage location;
(Col. 12, line 66, "…if an exception occurs when processing the host instructions…", col. 17, lines 59-62, "Placing the values from the target registers into the working registers may place the

address for the first of the target instructions (recover address from storage) which were running when the exception occurred in the working instruction pointer register…")

G) Recovering said system status from said second storage location;
(Col. 13, lines 2-4, "…the original state in the target registers at the last update (or commit) may be recalled (recover by recalling previous state which was saved in target / official registers) to the working registers…", col. 17, lines 50-54, "…code morphing software may cause the values retained in the official registers to be placed back into the working registers and any non-committed memory stores in the gated store buffer to be dumped (recover system status)…")

H) Repeating the execution of said first sequence of instructions beginning at said at least one of the instructions whereby each of said operations are executed in-order;
(Col. 13, line 8, "…executed in serial sequence…", col. 17, lines 65-66, "…are retranslated in serial order without any reordering or other optimizing…")

I) If said first sequence of instructions are executed and an exception is not detected:
(Col. 12, lines 57-58, "…one or a group of target instructions have been translated and run without error (exception not detected)", col. 18, lines 1-3, "…translated host instruction representing the target instructions is executed by the morph host and causes of does not cause (exception not detected) an exception to occur…")

J) Committing said status of said computer system to a register set;

(Col. 12, line 50, "...official memory state changes on a "committed"...", col. 18, lines 9-14, "...if a translation of a target instruction is executed without an exception being generated, then the state of working registers is transferred to the target registers and any data in the gated store buffer is committed...(commit to target register set)")

K) Repeating steps A) through F) for a second sequence of instructions.

(Col. 18, lines 6-7. "If no exception occurs...the next primitive function is run...", col. 18, lines 20-24, "...once a first target instruction of the series of instruction the translation of which generated an exception has been executed without generating an exception, the target instruction pointer points to the next of the target instructions..." Repeat translation, detection of exception during execution, and commit for each segment of instructions of target software.)

Per claim 18:

-registers set comprise a set of registers configured to mirror a foreign register set.

(Col. 12, lines 37-41, "The additional registers also allow the maintenance of a set of host or working registers for processing the host instructions and a set of target registers to hold the official state of the target processor...", col. 16, lines 6-21, "...along with an increased number of normal working registers...", col. 16, lines 41-49, "...when the processor begins executing the translated instruction sequence, the official target registers hold the values which could be held by the registers of the target processor for which the application was designed...After the morph hast has begun executing the translated instructions, however, the working registers hold values determined by the primitive operations of the translated instructions executed to that point."

Two sets of registers exist. They mirror each other. The working registers commit the values to the mapped target registers is execution proceeds without an exception.)

Per claim 19:

-speculative mode comprises the steps of Determining if an operation could generate an exception;

(Col. 13, lines 47-57, "...some exception are generated by host hardware...others are used by the code morphing software to detect failure (determine if speculative mode generates an exception) of various speculations...")

-Generating a diagnostic operand when an exception while in speculative mode;

(Col. 20, line 34-col. 21, line 56, "Two additional hardware enhancements help...The first of these is an abnormal/normal (A/N) protection bit (diagnostic operand is generated) stored with each address translation...Target memory operations within translations can be of two types, ones which operate on memory (normal) or ones which operate on a memory mapped I/O device (abnormal). A normal access which affects memory completes normally...the operations of an abnormal access...must be practiced in the precise order in which those operations are programmed...Without a means to distinguish memory from memory mapped I/O, it is necessary to treat all memory with the conservative assumptions used to translate instruction which affect memory mapped I/O. This severely restricts the nature of optimizations...the A/N bit is part of that other information and indicates whether the physical address is a memory address of a memory mapped I/O address...When the access type does not match the A/N protection, an exception occurs...If the comparison with the A/N bit in the TLB shows that the operation, however, affects an I/O device, then execution causes an exception to be taken..." An A/N protection bit (a diagnostic operand) is generated if the speculative mode execution incorrectly assumes an I/O memory access (exception in speculative mode), rather than an operation to alter memory. Also, col. 9, line 31-33, "...apparatus to...detect a failure of speculation on the nature of the memory being addressed...")

-Denote that said diagnostic operand is a speculative value;
(Col. 20, lines 62-64, "Prior art emulators lacked means to...detect a failure of speculation on the nature of the memory being addressed...", col. 21, lines 49-53, "...technique which uses the A/N bit to determine whether a failure of speculation has occurred as the whether an access is to memory or a memory-mapped I/O device may also be used for speculation regarding other properties of memory mapped addresses..." The A/N protection bit is stored with each address translation in a translation lookaside buffer (col. 20, lines37-39).)

-If said diagnostic operand is not modified when said diagnostic operand is changed to a non-speculative value, involving an exception.
(Col. 21, line 29-31, "...target memory reference is checked by comparing the access type presumed (normal or abnormal) against the A/N protection bit now in the TLB...When the access type does not match the A/N protection, an exception occurs. If the operation in fact affects memory, then the optimizing, reordering, and rescheduling techniques...were correctly applied...If the comparison with the A/N bit in the TLB shows that the operation, however,

affects an I/O device, the execution causes an exception…if a translation incorrectly assumes an I/O operation of an operation which actually affects memory (diagnostic operand not modified), execution causes an exception (involving an exception) to be taken…")

Per claim 20:

A method of optimizing binary translated code by extracting the parallelism inherent to foreign code during the binary translation process; said optimizing comprising the sequential reordering of operations and memory access without violating precise exception order in binary translated code executing on a host platform having explicit parallelism architecture, said method comprising the steps of:

(Col. 11, lines 3-7, "…code morphing software is software which translates the instructions of a target program to morph host instructions for the morph host and responds to exceptions and errors by replacing working state with correct target state…(precise exception order is maintained by executing a section of instructions at a time, detecting exceptions, then committing, before repeating with the next section of instructions)", col. 11, lines 45-46, "…allow the reordering and rescheduling (optimizing/ reordering) of primitive instructions generated by a sequence of target instructions…", col. 14, lines 15-17, "…the morph host is a very long instruction work (VLIW) processor which is designed with a plurality of processing channels…". Col. 14, lines 31-36, "…each of the primitive instructions which together make up a single very long instruction word is furnished (from target / foreign code) in parallel (extracting parallelism) with the other primitive instructions either to one of a plurality of separate processing channels…to be dealt with in parallel by the processing channels and memory…")

a) allocating a set of registers and a dedicated memory region for storing information regarding the location of a foreign register set in said register set;
(Col. 12, lines 31-58, "...The additional registers also allow...a set of target registers (foreign registers) to hold the official state of the target processor.....official memory state changes on a 'committed' side of the hardware gate (dedicated memory region) where these committed stores 'drain' to main memory (dedicated memory region)...")

b) designating a set of Recovery Points in said binary translated code where each Recovery Point in the binary translated code has correspondence with an instruction in the foreign code, and each Recovery Point is associated with a documentation set, saved on hard disk or in memory, that contains information where all foreign registers are located in the host registers in the optimized binary translated code;
(Col. 12, lines 59-69, "These updates are chosen by the code morphing software to occur on the integral target instruction boundaries." The code morphing software designates the recovery points. A section of code (a document set) is processed. Each section is a Recovery Point. Col. 12, lines 41-48, "The target (or shadow) registers are connected (foreign code registers) to their working register equivalents through a dedicated interface (interface contains information regarding the location of equivalent registers)..." Translated and executed code is processed in the working registers (holds documentation set), which are mapped to the target registers (foreign registers).)

c) responding to an exception between adjacent Recovery Points by re-invoked the interpretation of foreign instructions in sequential fashion by starting execution from the previous Recovery Point, i.e. foreign context (foreign registers and memory) must not be changed in the optimized binary translated code irretrievably.

(Col. 17, lines 59-62, "Placing the values from the target registers into the working registers may place the address of the first of the target instructions which were running when the exception occurred in the working instruction pointer register (start execution from the previous Recovery Point)…", col. 17, lines 64-66, "…the target instructions which were running when the exception occurred are retranslated (re-invoke the interpretation) in serial order (in sequential fashion) …")

Per claim 21:

-step of designating each memory write operation into the foreign memory space as a Recovery Point.

(Col. 20, lines 57-61, "Without a means to distinguish memory from memory mapped I/O, it is necessary to treat all memory with the conservative assumptions…" Kelly disclosed issues with translating memory access operations, an suggested an improved technique using the A/N bit. These instructions comprise a 'set of target instructions', or a Recovery Point. Col. 21, lines 49-53, "…uses the A/N bit to determine whether a failure of speculation has occurred as to whether an access into memory or a memory-mapped I/O device…", col. 22, lines 3-17, "…a set of instructions may first be translated as though it were to affect memory… " Translate, detect errors, commit and repeat.)

Per claim 22:

-designating at least one selected operation corresponding to an instruction in foreign memory space as a Recovery Point where said selected operation changes the state of the foreign memory space in an unrecoverable manner.

(Col. 12, lines 43-48, "…allows an operation called 'commit' (change state of foreign memory space / target memory by committing correctly executed results from a section of translated, executed code) to quickly transfer the content of all working registers to official target registers and allows an operation called 'rollback' to quickly transfer the content of all official target registers back to their working register equivalents…")

Per claim 23:

-maintaining the state of said computer system corresponding to the preceding Recovery Point until a next to occur Recovery Point is reached.

(Col. 17, lines 50-52, "In response to the detection of the target exception, the code morphing software may cause the values retained in the official register to be placed back into the working registers and any non-committed memory stores in the gated store buffer to be dumped (maintain state in target registers until a commit / 'next to occur Recovery Point' is reached)…")

Per claim 24:

-saving information sufficient to restore the contents of the foreign registers at the preceding Recovery Point.

(Col. 17, lines 50-52, "“In response to the detection of the target exception, the code morphing software may cause the values retained in the official register to be placed back into the working registers (information saved in target registers is sufficiently stores prior contents / preceding Recovery Point, for foreign registers) and any non-committed memory stores in the gated store buffer to be dumped…")

Per claim 25:

-saving step comprises the step of saving selected register names comprising foreign registers.

(Col. 12, lines 41-44, "The target (or shadow) registers are connected to their working register equivalents…that allows an operation called 'commit' (saving step, saves working / host registers to target / foreign registers if code executes correctly)…")

Per claim 26:

-renaming said selected registers for optimization of said binary translated code after passing said next to occur Recovery Point.

(Col. 12, lines 35-37, "Some of the additional registers allow the use of register renaming…", col. 18, line 56 - 59, "…translating the instructions, optimizing, reordering, rescheduling, caching, and executing each translation so that it may be rerun whenever that set of instructions need to be executed…" Each set of instructions marks another Recovery Point. Registers will be rolled back to the previous set of successfully executed instructions is an exception is detected. The 'next to occur Recovery Point', the next set of instructions to be processed, are optimized. Registers may be renamed.)

Per claim 27:

-marking each Recovery Point in the binary translated code with a "SetlP" operation.

(FIG. 8, linking process, col. 18, lines 59-61, "...translator also links the different translations to eliminate tin almost all cases a return to the main loop of the translation process." After a section of instructions is translated (a Recovery Point), the translator links the translated code, resetting the Instruction Pointer (SetIP operation) to point instead to the already translated code rather than retranslate a second identical piece of code. Also, col. 19, lines 23-26, "...the instruction pointer is updated in the target instruction pointer register...", col. 19, lines 47-53, "The update-instruction-pointer primitive operation preceding the branch tests the condition and determines that the loop back to the first translation is to be taken (instruction pointer has been set to look at the TLB rather that to retranslate code by going back to the main loop)...This causes the translator to look in the translation buffer to see if the ...address being sought appears there.")

Per claim 28:

-saving a host address of the wide instruction for the marked Recovery Point in a Recovery Point Register.

(Col. 17, lines 50-58, "If a target exception is generated during the running...that exception is detected...In response to the detection...the code morphing software may cause the values retained in the official registers (values for the Recovery Point are saved in the official registers)to be placed back into the working registers...")

Per claim 29:

ln a computer system adapted to executing a computer program comprising binary translated code, a method for reconstructing the cause of an exception comprising the steps of:

-temporarily preserving register data and system status information before executing instructions that will calculate a variable;

(Col. 12, lines 54-58, "The additional official registers and the gated store buffer allow the state of memory and the state of the target registers to be updated together once one or a group of target instructions have been translated and run without error." Register data is temporarily preserved in official (target) registers, and modified after executing instructions that calculate a variable.)

-invoking an exception handler prior to generating side effects;

(Col. 13, line 65-col. 14, line 2, "For the case where exceptions are used to detect failure of other speculations, such as whether an operation affects memory or memory mapped I/O, recovery (invoke and exception handler) is accomplished by the generation (prior to side effects) of new translations with different memory operations and different optimizations."

-responding to said exception with said exception handler using said temporarily preserved register data to redirect operation of said computer program

(Col. 13, line 65-col. 14, line 2, ", recovery (exception handler) is accomplished by the generation of new translations (redirect by returning to the beginning of code segment, using information stored in target registers for start address, for retranslation) with different memory operations and different optimizations."

Per claim 30:

-operands in the temporary registers are retained until reassigned by another operation.
(Col. 16, lines 56-63, "Once the translated host instructions begin, the values in the working registers (operands in the temporary registers) are whatever those translated host instructions determine the condition of those registers to be. If a set of translated host instructions is executed without generating an exception then the new working register values determined at the end of the set of instructions are transferred (operands are reassigned by commit operation)...")

Per claim 31:

-in response to an exception, the step of recalculating operands by re-executing the instruction to recover side effects.
(Col. 17, line 50-54, "In response to the detection of the target exception, the code morphing software may cause the values retained in the official registers to be placed back into the working registers (operands are reset back to initial values using values retained in the 'official' / target registers)..." Operands are recalculated after code is retranslated and re-executed. Col. 12, lines 6-11, "Then, for the case where the exception generated is a target exception, the target instructions causing the target exception may be retranslated one at a time and executed...As each target instruction is correctly executed without error, the state (includes operands) of the target registers may be updated (recalculated)...")

Per claim 32:

-step of allocating a portion of general-purpose registers in a register set to function as temporary registers.

(Col. 12, lines 31-58, "In order to overcome these limitations of the prior art, a number of hardware improvements are included...a gated store buffer and a large plurality of additional processor registers...The target (or shadow) registers are connected to their working register equivalents (temporary registers) through a dedicated interface...")

Per claim 33:

-step of allocating a memory region for storing information regarding the location of a foreign register set in said register set.

(Col. 12, lines 31-35, "In order to overcome these limitations of the prior art, a number of hardware improvements are included...a gated store buffer and a large plurality of additional processor registers...", col. 12, lines 41-48, "The target (or shadow) (foreign registers) registers are connected (memory is allocated for the foreign register set) to their working register equivalents through a dedicated interface...")

Per claim 34:

A computer system adapted for executing a computer program comprising binary translated code comprising:

-a first memory for storing foreign code / a second memory for storing optimized binary translated code;

(Target Application Memory (registers after a commit) / Host Application Memory (working registers prior to a commit), Col. 11, lines 3-7, "...while code morphing software is software which translates the instructions of a target program to morph host instructions...and responds to exceptions and errors by replacing working state (a second memory) with correct target state (a first memory)..."

-a recovery point register for storing the address of a selected instruction in said binary translated code;

(Col. 12, lines 31-58: Target registers holds recovery point information, including the address of a selected instruction.)

-a processor for executing binary translated code and adapted for detecting the occurrence of an exception, said processor coupled to said recovery point register and to said first and second memory;

(Col. 9, lines 31-33, "...apparatus to both detect a failure of speculation on the nature of the memory being addressed, and apparatus to recover from such failures...", col. 11, lines 3-7, "...translates the instructions of a target program to morph host instructions...and responds to exceptions and errors by replacing working state with correct target state (recovery point register) when necessary so that correct retranslations occur." Recovery point information held in target registers may be used to rollback working registers to a recovery point, upon detecting an exception)

-an exception handler, coupled to said processor, for recovering the address stored in said recovery point register in response to detection of an exception by said processor, said address corresponding to one instruction in said foreign code;
(Col. 11, lines 5-6, "...responds to exceptions and errors by replacing working state with correct target state...", col. 12, lines 41-48, "The target (or shadow) registers (holds address of foreign code) are connected to their working register equivalents...allows an operation called 'rollback' to quickly transfer the content of all official target registers back to their working register equivalents (reset working registers)...", col. 17, lines 59-62, "Placing the values from the target registers into the working registers may place the address of the first of the target instructions (recover address – address corresponding to one instruction in said foreign code) which were running when the exception occurred (in response to exception) in the working instruction pointer register.")

-a set of documentation stored in said second memory for preserving register data and system status information of the state of said computer system prior to detection of said exception;
(Col. 12, lines 31-58: Prior to the detection of an exception, the most current register data and system status information is preserved in the working registers.)

-means for translating said foreign code to obtain sequential binary code corresponding to a portion of foreign code beginning at said one instruction in said foreign code.
(Col. 17, lines 64-66, "…target instructions which were running when the exception occurred are retranslated in serial order without any reordering of other optimizing…")

Conclusion

12. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (703) 305-4564. The examiner can normally be reached Monday through Thursday, from 7:00 A.M. to 5:30 P.M. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on (703) 305-4552.

The fax phone number is (703) 872-9306 for regular communications and for After Final communications. Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Mary Steelman [illegible]
06/30/2004

[illegible]
TUAN DAM
SUPERVISORY PATENT EXAMINER